A Software Radio-based UHF RFID Reader for PHY/MAC Experimentation

Michael Buettner
University of Washington
buettner@cs.washington.edu

David Wetherall
University of Washington
djw@cs.washington.edu

***Abstract*—We present the design and evaluation of a flexible UHF RFID reader that enables new PHY/MAC designs to be prototyped and evaluated. Our reader is built using the USRP software radio platform in conjunction with software we developed in the open-source GNU Radio framework. We believe it is the first inexpensive tool that readily enables changes to the physical and MAC layer of RFID systems. We evaluate our reader and show that it can inventory commercial tags out to 6 meters, which approximates the range of a commercial reader with comparable transmit power. We then show two applications of our reader. The first evaluates the real-world performance of the EPC frame selection algorithm and finds that it performs better than expected. Second, using the Intel WISP programmable RFID tag, we implement and evaluate an extension to the Gen 2 standard that results in up to a five-fold increase in sample rate for streamed sensor data.**

I. Introduction

Radio Frequency IDentification (RFID) is an emerging wireless technology that allows small, inexpensive computer chips to be remotely powered and interrogated for identifiers and other information. While there are many kinds of RFID, e.g., HF RFID in credit cards, recent advances in RFID have largely focused on passive UHF RFID as standardized by the EPC Class-1 Generation-2 (Gen 2) specification in 2004 [1].

The application space for UHF RFID is rapidly expanding, and the ability to experiment with realistic systems is becoming increasingly important. For instance, the reliability and performance of RFID readers is of interest, especially in dense, fine-grained settings (such as item-level tracking) and for demanding applications (such as searching over the states of objects in a ubicomp application). However, there is very little published information on low-level RFID performance in these settings. Prior work on the topic concludes that current reader systems suffer various performance degradations and there is ample opportunity for improvement [2], [3]. Additionally, privacy and security issues are central to any application that tracks objects and people directly or indirectly. Yet almost all work on RFID security using lightweight cryptographic schemes has been done via paper designs and analysis rather than experimentation; we are aware of only one exception [4].

Looking ahead, as RFID devices advance to include computation, storage and sensing [5], [6], [7], alternative MAC and PHY layer designs will inevitably emerge and need to be evaluated. This is because the Gen 2 protocol was designed with simple object identification in mind and is a poor fit for gathering sensor data or supporting complex interactions with devices [8].

The dearth of low-level experimentation with RFID systems is a consequence of the current lack of tools available to researchers. Existing RFID readers are generally black box systems which provide only limited configuration and return high-level results simply indicating tags that are in range. They do not provide the flexibility to observe or modify the MAC or PHY layer behavior. This makes the study of existing protocols difficult, let alone experimentation with alternative designs. High-end RFID test equipment is available in the form of protocol analyzers and systems have been built to measure tag performance [9]. While well suited for conformance testing, they are closed source, expensive (>$80K), and generally do not allow complete control over the PHY and MAC layers.

In this paper, we present what we believe is the first open-source platform for low-level UHF RFID experimentation that gives users control of the PHY and MAC layers. It consists of the Universal Software Radio Peripheral (USRP) and software we have developed using GNU Radio that implements a EPC Class-1 Generation-2 reader. Because all signal processing is performed on a standard Linux PC, MAC and PHY functionality can be modified simply by re-writing user-level software. Our platform provides a high degree of flexibility while costing less than $1500.

We make two main contributions in this work. First, we present the design and implementation of our reader and show how a low-cost SDR platform and open-source software can be used to create a powerful tool for UHF RFID experimentation. We describe how we overcame the limitations inherent to the hardware which made implementing the reader difficult, and evaluate the reader performance, finding that it can inter-operate with standard Gen 2 tags at a range of up to 6 meters. This

978-1-4244-9606-8/11/$26.00 ©2011 IEEE

closely approximates the range of a commercial RFID reader when using the same transmit power of 27 dBm.

Second, we present two applications of our reader that demonstrate its usefulness as a research tool. The first application uses our reader and commercial RFID tags to study the performance of the frame size selection algorithm described in the EPC Gen 2 standard. We find that the algorithm's performance is within 7% of the maximum rate achieve using an oracle based algorithm. Also, because of RF capture, the algorithm achieves a 44% slot efficiency which is well above the the best expected efficiency of 37% using the standard collision model. This has implications for how such algorithms are designed and analyzed.

The second application shows that, when paired with an equally flexible RFID tag, our reader can be used to develop entirely new MAC protocols. We demonstrate this capability by prototyping an extension to the Gen 2 standard to accommodate high-rate streaming data for sensor-equipped RFID tags. Through experiments we show that our extension can provide a 5X improvement in sampling rate compared to the normal Gen 2 mechanism for reading data from tags.

The software described in this document is open source and available for download from the Comprehensive GNU Radio Archive[1]. We hope that by demonstrating the potential of our reader for experimenting with RFID systems, the RFID community will see ways in which the platform can be useful for their work.

II. Goals and Challenges

We set out to develop a Gen 2 RFID reader that could communicate with commercial tags while giving researchers complete control over the physical and MAC layer protocols. Such flexibility would enable the detailed study of current RFID systems, and provide a platform for experimentally validating proposed protocols.

To provide a low barrier of entry for researchers, our reader needed to be based on common, low-cost, off-the-shelf components and not depend on specialized or custom-built hardware. Additionally, we wanted the platform to be accessible to users without FPGA expertise or a deep background in signal processing.

To meet these goals, we built our system using the Universal Software Radio Peripheral (USRP) and the GNU Radio signal processing toolkit. The USRP is a general-purpose RF front-end for software radio development with an effective complex sampling rate of 8 Msps. It interfaces via USB with a standard Linux PC where essentially all signal processing is performed. Daughterboards convert RF signals to and from baseband where an ADC and DAC are used to transform the signal from analog to digital and vice versa.

[1] www.cgran.org

On the host, signal processing is implemented within the GNU Radio framework, a software library and runtime system developed as a counterpart to the USRP. Applications that tie signal processing blocks together are implemented as user-level Python programs and signal processing blocks are implemented in C++.

Using the USRP and GNU Radio for our transceiver is ideal in terms of flexibility. However, this flexibility comes at the cost of performance, and the limitations of the platform made meeting the Gen 2 timing requirements difficult. First, the USB interface has been shown to introduce system latency on the order of tens of milliseconds [10], while Gen 2 tags will ignore reader commands if the latency is greater than 500 μs. Meeting the Gen 2 timing requirements required modifications to the GNU Radio scheduler and USB subsystem, and careful implementation of the signal processing chain.

Second, because GNU Radio is implemented on general purpose hardware, the transmit and receive chains are decoupled by a series of buffers within both GNU Radio and the Linux kernel. The result is that when a signal is generated by GNU Radio, it is difficult to know how long it will take before that signal is transmitted by the radio hardware or how long it will be before it is processed by the receive chain. This made it difficult to assure that the reader transmits messages at precisely the right times. We will discuss how our design overcomes these challenges later in the paper.

III. Applications

Our platform provides a foundation for researchers to conduct UHF RFID research. As we have implemented a Gen 2 compliant reader, it can be used with commercial tags to experiment with modifications to the Gen 2 PHY and MAC layers. A flexible PHY implementation lets researchers readily control reader behavior such as what frequency is used, how and when signals are transmitted, and how tag responses are decoded. Previous work has shown that such control would be useful to study fading [2], power transfer [11] and alternative tag decoding strategies [12].

A flexible MAC implementation means that Gen 2 MAC parameters can be readily manipulated to understand their impact on reader behavior, and anti-collision algorithms can be experimentally validated; such research has generally suffered from the opacity of commercial readers [2], required specialized equipment [13] or was done via analysis and simulation [14].

Because our reader functionality is implemented as user-level software, our platform can be used as a starting point to experiment with non-Gen 2 UHF backscatter devices. New PHY layers for UHF RFID have been proposed such as CDMA [15] and QAM [16], and experimental UHF backscatter platforms are emerging [7], [6].

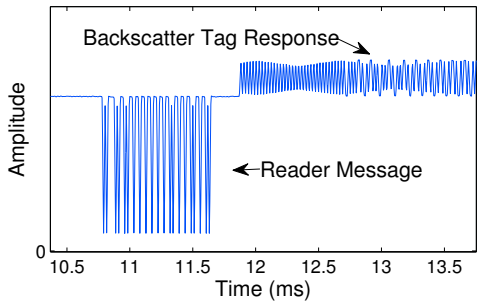

Fig. 1. Reader message and tag response

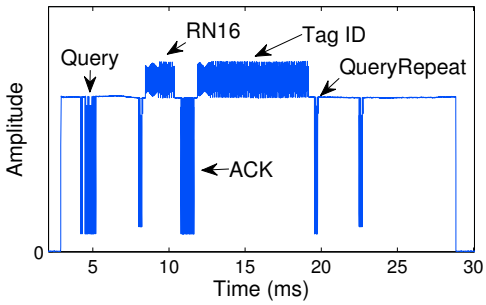

Fig. 2. Message exchange for a Query round

Our reader can be easily modified to debug and evaluate such prototypes, and to explore completely new ways of interacting with UHF backscatter devices beyond Gen 2. One platform that is widely available and has been used with standard RFID readers is the Intel WISP [5]. The WISP is a fully programmable RFID tag with sensors that implements the Gen 2 MAC and PHY layers in software. When used in conjunction with our reader, it can be used to implement and evaluate novel MAC protocols as we show later in this paper.

IV. Gen 2 RFID Background

In this section, we briefly overview the Gen 2 PHY and MAC layers that are implemented by our reader.

A. Gen 2 Physical Layer

When communicating with tags a reader must transmit a continuous RF wave (CW); tags power themselves by harvesting the energy in this RF signal. Down-link communication uses On/Off Keying where bit boundaries are indicated by brief pulses of zero amplitude in the CW, and Pulse Interval Encoding (PIE) where the time between pulses differentiates a zero or a one.

Strictly speaking, passive RFID tags do not transmit energy when communicating with a reader. Instead, they manipulate how well they reflect (backscatter) the incident CW. This reflected signal, though weak, is received by the reader which decodes the modulated data. The reader specifies what link frequency tags use for the uplink ranging from 40 kHz to 640 kHz.

Figure 1 shows communication between our reader and a tag as viewed from our reader. The CW results in the DC offset of the received waveform, with the series of low amplitude pulses being a reader transmission. The backscattered tag response can be clearly seen as a combination of the incident CW and the reflected CW from the tag. It should be noted that the signal seen in the figure was captured with the USRP antenna placed inches from the RFID tag. This accounts for the prevalent backscatter signal.

B. Gen 2 MAC Layer

Gen 2 tags decode reader transmissions using a simple edge detector that detects the conspicuous zero amplitude pulses in the CW. However, tags are unable to decode, or even detect, the backscattered signals of other tags. Consequently, the Gen 2 MAC protocol is based on Framed Slotted Aloha [17] which was designed to operate in a context where transmitting nodes cannot hear each other.

The general model of RFID is that readers are continuously "inventorying", i.e., looking for tags that are in range. An inventory round begins with the reader transmitting a *Query* command that indicates the number of slots in the frame. The number of slots in a frame is a power of two in the range [1, 32768]. After receiving a *Query*, tags randomly choose a slot in which to reply, and transmit a 16-bit random number (*RN16*) in that slot. If the reader receives an *RN16* in a slot, it *ACKs* the *RN16* and the tag that transmitted it replies with its 96-bit identifier. Tags that collide in a slot are not *ACKed* and respond again after the next *Query*.

Figure 2 shows a message exchange between our reader and a single tag. In the example, our reader powers up and transmits a *Query* message that specifies four slots in the frame. The first slot immediately follows the *Query* command, and is empty in this example. The second reader command is a *QueryRepeat* which indicates the beginning of a new slot. In this case, the tag had chosen the second slot and so it transmits its *RN16* in response to the *QueryRepeat*. It is then *ACKed* by the reader, and transmits its *ID*. Because the reader specified four slots in the *Query*, it sends two additional *QueryRepeats* looking for any remaining tags. In this case, there is only a single tag so the last two slots are empty. In practice, readers perform a series of *Query* rounds while varying the number of slots in the frame based on how many tags are responding; tags stop responding once they transmit their *ID*. This continues until no tags respond, at which point the reader powers down briefly.

V. Reader Architecture

Our reader hardware consists of a USRP1 equipped with two RFX900 daughterboards, one for transmit and

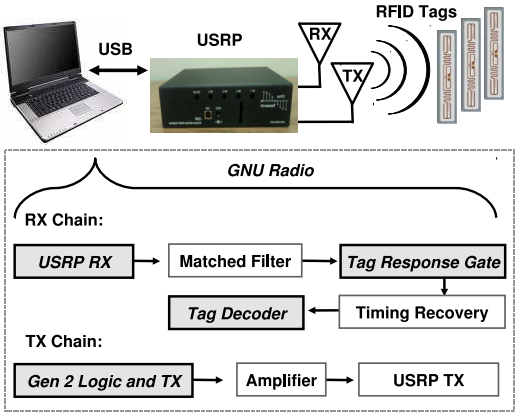

Fig. 3. Block Diagram of our Reader Architecture

one for receive. The GNU Radio based software architecture uses standard blocks provided by the toolkit, along with custom blocks we developed that implement the Gen 2 specific functionality. In this section, we give an overview of the architecture and describe how it is configured and what data it produces.

A. Overview

Figure 3 shows a simplified block diagram of our reader architecture with shaded blocks being those we created or modified. The *Gen 2 Reader* block implements the Gen 2 state diagram and generates the On/Off keyed reader commands for transmission. Commands are then sent to a configurable amplifier and sent to the USRP hardware for transmission.

The first GNU Radio block in the receive chain is the *USRP RX* block which pulls received samples in from the Linux kernel and feeds the flowgraph. The samples are first passed through a matched filter which is configured to maximize the signal-to-noise ratio (SNR) of tag transmissions. The filtered signal is then sent to the *Tag Response Gate* which acts as a signal gate, only ungating the incoming signal when a tag response must be decoded. This reduces computation as clock recovery and tag decoding are executed only when a tag response is expected, which reduces overall system latency.

When ungated, the signal is passed through a clock recovery block which resamples the tag response and outputs one sample per subcarrier cycle. The *Tag Decoder* block then detects the preamble and demodulates the ASK modulated tag transmission. After the response is decoded, a message is sent (via function call) to the *Gen 2 Reader* block along with the decoded bits; this triggers the reader to send out the next command.

B. Configuration and Output

As our reader is implemented in user-level software, it can be arbitrarily reconfigured by the user. We tried to design our system such that logical functions are cleanly isolated in different blocks. Demodulation, for example, is implemented in a single block, though the filter and timing recovery blocks may need to be reconfigured to support different modulations. Changes to the MAC are easily achieved by adding to or modifying the state diagram defined by the *Gen 2 Logic and TX* block. In addition, we attempted to expose the most common parameters for users that want standard behavior from the reader but with greater parameterization and visibility. For example, all Gen 2 MAC parameters can be configured, such as the session and up-link encoding.

Our reader generates two types of output files. The first is a trace of reader commands and tag responses with millisecond resolution along with the SNR for tag responses and empty slots. The second type of output is a raw trace of the signal that is output by any (or all) blocks. For example, Figures 1 and 2 were generated by plotting the amplitude of the signal output by the *Matched Filter* block. This eases the task of debugging the system or adding new functionality as the signal can be inspected at each stage.

VI. Meeting the Gen 2 Timing Constraints

The Gen 2 protocol dictates strict timing requirements for reader and tag transmissions. If reader commands are transmitted too soon or too late after a tag transmission, the tag will ignore the command. The flexibility of the USRP and GNU Radio come at the cost of high and unpredictable latency. This section describes the techniques we used to overcome these limitations.

A. Reducing System Latency

The first timing related challenge we overcame was responding quickly enough to tag responses, for example when sending an ACK after decoding an RN16. This deadline is measured from the last edge of the tag response to the first edge of the reader response, and is defined as 20 times the duration of one tag carrier cycle, e.g., if a reader specifies a 40 kHz uplink frequency the system latency must be less than 500 μs. Reader messages transmitted after this deadline are ignored by the tag. A main source of system latency is the bandwidth between the USRP and the host. With advances in USRP hardware, for example GigE instead of USB 2.0 on newer platforms, overall latency will be reduced to some extent. However, using our software with the USRP1 required careful implementation to meet the protocol requirements.

Our initial reader implementation had a latency of more than 20 ms. We reduced this latency in two main ways. First, a large fraction of the latency was because the USB interface transferred samples to the receive chain in chunks of 4096 samples. When using a 250 ksps input signal, one chunk represented more than 16

ms worth of signal and introduced, on average, 8 ms worth of latency. By reducing this block size to the minimum allowed by the Linux kernel (128 samples), and using a high rate input stream which is decimated by the *Matched Filter* block we achieved latencies of around 400 μs.

To further reduce latency, we carefully controlled how blocks are scheduled in the flow graph. Once a tag response is detected by the *Tag Decoder* block, upstream blocks process only the number of samples that represent the remainder of the tag response. Upstream processing is then stalled until the *Tag Decoder* finishes decoding the response and the next reader command generated by *Gen 2 Logic and TX* block has been transferred to the Linux kernel. This is in contrast to normal GNU Radio scheduling where blocks are scheduled in a round robin fashion with the result that upstream blocks spend time processing the CW following the tag response before the next reader command is transmitted. These techniques brought the latency of our system down to below 250 μs. This is well below what is needed to interoperate with commercial Gen 2 tags, and sufficient for up-link frequencies up to 80 kHz.

B. Precise Command Timing

The last timing related difficulty relates to the transmission of the CW by the reader. When the USRP has no messages to send it transmits a zero amplitude signal by default, so our reader software must transmit the CW explicitly. Because of the transmit buffers in GNU Radio and the Linux kernel, when the reader inserts a command into the tail of the TX buffer it will take an unknown amount of time to be transmitted because it will be behind some number of CW samples; this both increases latency and makes precise timing difficult.

To overcome this problem, the *Gen 2 Logic and TX* block calculates precisely how many samples worth of CW needs to be transmitted before the next reader command must happen. This can be determined as the tag responses are of a known length and the inter-command timing is well specified. Immediately after sending a reader command to the Linux kernel, the block injects the right number of CW samples into the transmit buffer. As long as the next command is not inserted too late, it is guaranteed to be transmitted within the timing bounds of the protocol.

VII. Gen 2 Reader Evaluation

In this section, we evaluate our reader to show how well it performs, with a focus on the degree to which the platform can be used for further research.

A. Experimental Setup

For all experiments in this evaluation, we use Alien "Omni-Squiggle" tags attached to a sheet of poster board with the tags being approximately one meter off the ground. Experiments with our reader use a 40 kHz Miller-2 modulated uplink. Our reader has an output power of 27 dBm as measured by a power meter, and uses a bi-static configuration with two Cushcraft S9028PCLW antennas (one for RX and one for TX) on one meter stands. The experiments were conducted in a standard office setting, and we attempted to produce ideal conditions for the reader; i.e. line of sight with minimal objects in the area. As a comparison, we use an Impinj Speedway reader and related experiments are conducted without moving the tags.

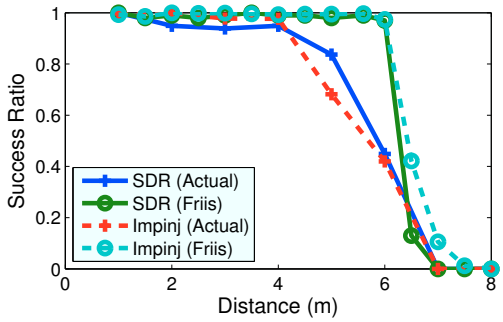

Fig. 4. Performance of our SDR reader and a commercial reader

B. Reader Performance

For our reader to be useful it must be able to read tags reliably at a range of at least a few meters. We first performed an experiment reading a single tag at increasing distances. To factor out the effects of multipath and to gather data in a more idealized setting, we repeated the experiment keeping the tag at one meter and reducing the transmit power of the reader to approximate distance based on Friis equation. At each distance we performed 1000 Query attempts, with each Query having only a single slot. As a comparison, we read the same tag using the Impinj reader at the same distances and attenuations. As we cannot get fine grained data from the Impinj reader, we approximate its success ratio by comparing the rate achieved at a given distance against the maximum rate seen for all distances.

Figure 4 shows that our reader can read a commercial tag from up to 6 meters away, and has a more than 95% success rate out to 4 meters. In the more idealized case, we saw nearly perfect success and then a sharp drop-off at what approximates 6 meters.

Compared to the Impinj reader, our platform performs quite well. The range of the two readers and success ratios were nearly identical in both the real world and idealized settings. Though the Impinj reader cannot be configured to use the same uplink settings as our reader, we configured it to use the most robust settings available. While not an apples to apples comparison, our results

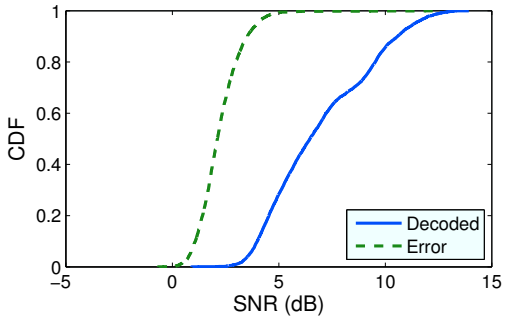

Fig. 5. Signal to noise ratio needed to decode tag responses

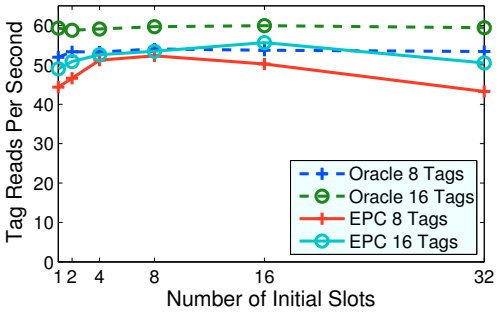

Fig. 6. Frame selection algorithm performance (reads per second)

show that our reader approximates the range of a commercial reader given comparable transmit power.

To evaluate how well our receive chain works, we repeatedly queried a single tag placed one meter from the reader while injecting noise into the receive chain. This was done in GNU Radio by combining the received signal with a Gaussian noise source immediately after the *Matched Filter* block. We increased the amplitude of the noise source incrementally until the tag could no longer be read; 500 Queries were sent at each of 12 noise levels, and SNR was measured at the receiver.

Figure 5 shows the cumulative distribution function of the SNRs for tag responses that were correctly decoded and those that contained bit errors. More than 95% of the responses that had errors had an SNR of less than 4 dB, i.e., responses with an SNR of greater than 4 dB were generally decoded successfully. This is corroborated by the fact that only 10% of the successfully decoded responses had an SNR less than 4 dB.

Considering we use no specialized hardware in our implementation, we are pleased with the performance of our reader. In particular, we feel that a range of 6 meters is sufficient for many research tasks. Commercial readers generally have hardware to isolate the receive chain from the *CW* signal, use sharp cut-off filters to suppress all but the tag response, and transmit at higher power. However, our goal is to provide a low-cost, low-complexity solution for researchers interested in RFID. As such, we leave evaluating the benefits of additional hardware to future work.

VIII. Studies Using Our Reader

In this section, we describe two studies we performed that demonstrate how our platform can be used. First, we use our reader and commercial RFID tags to study the performance of the frame selection algorithm described in the EPC Gen 2 standard. Second, we combine our reader with the Intel WISP programmable RFID tag to prototype a MAC protocol for rapidly streaming sensor data.

A. Performance of the EPC Frame Selection Algorithm

A key factor determining reader performance is accurate frame size selection. Slotted Aloha based protocols have a well-known maximum throughput of $e^{-1} = 0.368$ which is achieved when the number of slots is equal to the number of transmitters. In the case of Gen 2, this means the maximum throughput is where approximately 37% of the slots have a single tag response and the tag can be read. Because the tag population size is generally unknown, the EPC Gen 2 standard presents an algorithm for adaptively changing the frame size to match the population size. This algorithm has been studied through analysis and simulation [18], and alternative algorithms for choosing frame size has been an active area of research [14].

To see how the EPC algorithm works in practice, we placed commercial tags one meter from our reader and evaluated the performance of two algorithms. The first algorithm is that described in the EPC standard [1] where the frame size is updated incrementally depending on if zero, one, or more than one tag response is detected in a slot. The second is an Oracle algorithm where the size of the next frame is chosen based on the number of remaining tags which we know a priori.

Figure 6 shows the throughput in terms of tag reads per second for both 8 and 16 tags. We ran the experiment while varying the initial number of slots in the frame from 1 slot to 32 to determine how this impacts performance. For each frame size, 500 read cycles were performed, each of which continued until all tags were read.

For the Oracle algorithm, the 60 reads per second throughput for 16 tags is around 12% higher than for 8 tags. This is because the overhead of the initial *Query* and power down period are amortized when there are more tags. Performance for the Oracle algorithm is independent of starting frame size as it immediately selects the "right" size for the next frame.

The performance of the EPC algorithm is highly dependent on initial frame size, a point generally overlooked in the literature. When the initial frame size

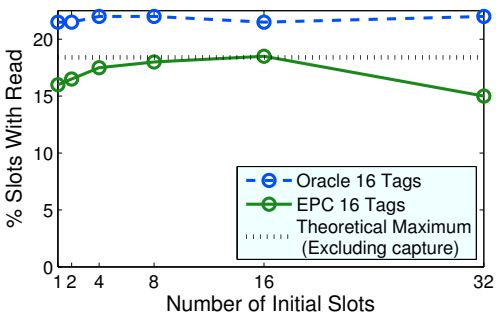

Fig. 7. Frame selection algorithm performance (slot efficiency)

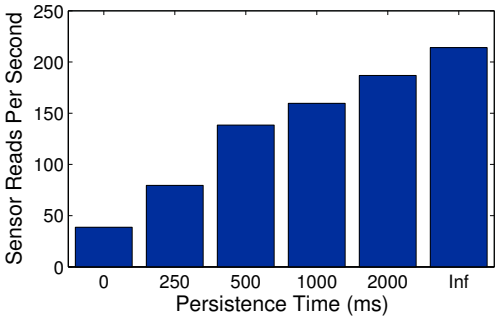

Fig. 8. Sensing rate for varying *Handle* persistence times

is optimal for the number of tags, the throughput is 97% and 93% of the maximum rate for the 8 tag and 16 tag cases respectively. Even if the frame size is underestimated by 1 frame size option, throughput is still above 95% and 89% of the maximum for the two cases. For many applications, such as inventorying items at a dock door, a rough estimate of the number of tags may be available and the algorithm can perform quite well.

Frame size selection algorithms are generally evaluated by how closely they approach the 37% theoretical maximum of successful slots, and many algorithms can achieve around 30-33% [19]. Figure 7 shows the percentage of slots where a tag was read by our reader for the 16 tag case. In our experiment, the EPC algorithm achieved the theoretical maximum of 37% when the initial number of slots was optimal, and the Oracle saw more than 44% of the slots with a successful tag read. The Oracle exceeds the theoretical maximum due to the capture effect, where a stronger tag response can be decoded if it collides with a weaker one. This suggests that algorithms that explicitly consider capture, and physical layer techniques that make capture more likely, have the potential to increase reader performance. Though demonstrating these techniques is out of scope for this paper, our platform provides a means by which this type of research can be conducted.

B. A New Protocol for Streaming Sensor Data

The EPC Gen 2 specification is well designed for inventorying applications, where tags store an identifier and possibly a small amount of data such as the date of manufacture. However, activity recognition systems using sensor equipped RFID tags [20] work by analyzing streaming sensor data from multiple tags. Unfortunately, the Gen 2 mechanism for reading data from tags incurs the overhead of tag singulation for every datum. This is acceptable when data reads and writes are done rarely, but becomes prohibitive when reading data is the primary activity of the application. We extend the Gen 2 protocol to enable efficient gathering of sensor data.

To read data from a tag according the the Gen 2 standard, the tag is first "singulated" using the protocol described in Section IV-B. After receiving the ID but before sending the next *QueryRepeat*, the reader requests a 16 bit *Handle* from the tag. The reader then transmits a *READ* command containing the *Handle* along with a memory location and data length; the tag then backscatters the specified data. *Handles* are valid for only a single transaction[2], and effectively expire when the next *Query* or *QueryRepeat* is received.

We extend the Gen 2 protocol by allowing *Persistent Handles*. With this, tags remember their last transmitted *Handle* for a period of time, and if they receive a *READ* command containing that handle they immediately backscatter the indicated data. Gen 2 tags already store state such as the *Selected* and *Inventoried* flags for a duration determined by the *Session* flag indicated by the reader; our extension simply requires that the *Handle* be maintained along with these other state variables. However, the reader should occasionally perform the singulation process both to refresh tag *Handles* and to search for any new tags.

We implement the EPC mechanism and our protocol extension using our reader and the Intel WISP. We deployed two WISPs one meter from our reader antenna and measured how quickly data could be read using the two mechanisms. Figure 8 shows the total number of sensor reads across both tags for varying persistence times. A persistence time of zero is the EPC mechanism where singulation takes place, one datum is read from each WISP, and the singulation process is started again. This approach sees approximately 40 sensor reads per second. A persistence time of infinity is where our reader singulates the tags a single time to gather their *Handles*, and then continuously sends *READ* commands that alternate between the two tags. This approach achieves around 215 sensor reads per second, a more than a five-fold increase. For intermediate persistence times our reader must perform singulation to refresh the *Handle*. Even when singulation is performed every 250 ms, overhead is reduced such that a sensing

[2]A single transaction may consist of multiple data read and writes.

rate of nearly 80 samples per second is achieved, more than double the rate achieved using the EPC mechanism.

Our extension is only a first step towards efficient sensor streaming, and the implications of *Persistent Handles* require more consideration. We leave this to future work. However, the study shows the potential our reader has for evaluating new protocol designs in real-world scenarios, particularly when paired with a programmable RFID tag such as the WISP.

IX. Related Work

There is a variety of existing experimental RFID platforms seen in the literature. [21] provides a custom fabricated UHF platform where signal processing is implemented in an FPGA, and [22] is a free hardware design for HF RFID. In contrast, our work uses standard off-the-shelf hardware, and the Gen 2 protocol is implemented as open-source software that is available to the research community and runs on a standard Linux PC.

There has also been work using the USRP to study RFID systems. Generally these have been monitors such as [23] that monitored subway card transmissions and our previous work [2], [3] that monitor commercial Gen 2 systems. The closest work to ours, though only a receiver, is [12] that uses a USRP to experiment with decoding concurrent HF tag transmissions. Also [24], that builds on our reader system to implement distributed RFID sensing. We hope our platform will be useful as a foundation for just this type of research.

X. Conclusion

We present a Gen 2 UHF RFID reader developed using the USRP and GNU Radio which can communicate with commodity RFID tags at up to 6 meters. As the complete Gen 2 protocol is implemented in user-level software, our reader gives a high degree of flexibility with respect to both the MAC and PHY layers of the system. This flexibility enables low-level RFID research which is not possible using commercial platforms.

To operate with commodity tags, the system latency of our reader must be below 500 μs. We present techniques that reduce the latency of the USRP and GNU Radio so that we can reliably meet this timing constraint. We then evaluate our reader performance and compare this to a commercial reader with comparable power, and show that we achieve approximately the same range while requiring no specialized hardware. We also discuss potential applications of our reader, and present two studies that show how our system can be used to experiment with commercial tags and also to prototype entirely new protocols when paired with a programmable tag. Our software defined Gen 2 reader provides a flexible and capable development platform that extends the application space of UHF RFID systems.

References

[1] EPCglobal, "EPC radio-frequency identity protocols class-1 generation-2 UHF RFID protocol for communications at 860 MHz-960 MHz version 1.0.9," 2005.

[2] M. Buettner and D. Wetherall, "An empirical study of UHF RFID performance," in *Proc. 14th ACM Int. Conf. Mobile Computing and Networking*, 2008, pp. 223–234.

[3] M. Buettner and D. Wetherall, "A "Gen 2" RFID monitor based on the USRP," *SIGCOMM Computer Communication Review*, pp. 41–47, June 2010.

[4] H.-J. Chae *et al.*, "Maximalist cryptography and computation on the WISP UHF RFID tag," in *Proc. Conf. RFID Security*, 2007.

[5] A. Sample *et al.*, "Design of an RFID-based battery-free programmable sensing platform," *IEEE Transactions Instrumentation and Measurement*, pp. 2608 –2615, 2008.

[6] D. Yeager *et al.*, "A 9.2a gen 2 compatible UHF RFID sensing tag with -12dbm sensitivity and 1.25vrms input-referred noise floor," *IEEE Int. SolidState Circuits Conf.*, pp. 52–53, 2010.

[7] M. Reynolds and S. Thomas, "The blue devil WISP: Expanding the frontiers of the passive RFID physical layer," *presented at the WISP Summit Workshop*, 2009.

[8] M. Buettner *et al.*, "Revisiting smart dust with RFID sensor networks," in *Proc. 7th ACM Workshop Hot Topics Networks*, 2008.

[9] P. Nikitin and K. Rao, "Labview-based UHF RFID tag test and measurement system," *IEEE Transactions Industrial Electronics*, 2009.

[10] T. Schmid *et al.*, "An experimental study of network performance impact of increased latency in software defined radios," in *Proc. 2nd ACM Int. Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization*, 2007, pp. 59–66.

[11] M. Trotter and G. Durgin, "Survey of range improvement of commercial RFID tags with power optimized waveforms," in *Proc. IEEE Int. Conf. RFID*, 2010, pp. 195 –202.

[12] D. Shen *et al.*, "Separation of multiple passive RFID signals using software defined radio," in *Proc. IEEE Int. Conf. RFID*, 2009, pp. 139 –146.

[13] P. Nikitin and K. Rao, "Effect of Gen2 protocol parameters on RFID tag performance," in *Proc. IEEE Int. Conf. RFID*, 2009, pp. 117 –122.

[14] C. Floerkemeier, "Bayesian transmission strategy for framed aloha based RFID protocols," in *Proc. IEEE Int. Conf. RFID*, 2007, pp. 228 –235.

[15] C. Mutti and C. Floerkemeier, "CDMA-based RFID systems in dense scenarios: Concepts and challenges," in *Proc. IEEE Conf. RFID*, 2008, pp. 215 –222.

[16] S. Thomas and M. Reynolds, "QAM backscatter for passive UHF RFID tags," in *Proc. IEEE Int. Conf. RFID*, 2010, pp. 210 –214.

[17] L. G. Roberts, "Aloha packet system with and without slots and capture," *SIGCOMM Computer. Communication Review*, pp. 28–42, April 1975.

[18] Y. Kawakita and J. Mitsugi, "Anti-collision performance of Gen 2 air protocol in random error communication link," in *Int. Symp. Applications and the Internet Workshops*, 2006.

[19] C. Floerkemeier and M. Wille, "Comparison of transmission schemes for framed ALOHA based RFID protocols," in *Int. Symp. Applications and the Internet Workshops*, 2006.

[20] M. Buettner *et al.*, "Recognizing daily activities with RFID-based sensors," in *Proc. 11th Int. Conf. Ubiquitous computing*, 2009, pp. 51–60.

[21] C. Angerer *et al.*, "A flexible dual frequency testbed for RFID," in *Proc. 4th Int. Conf. Testbeds and Research Infrastructures for the Development of Networks & Communities*, 2008, pp. 3:1–3:6.

[22] "Open PCD," [Online] Available: http://www.openpcd.org.

[23] R. Ryan *et al.*, "Anatomy of a subway hack," [Online] Available: tech.mit.edu/V128/N30/subway/Defcon_Presentation.pdf, 2008.

[24] D. De Donno *et al.*, "Challenge: towards distributed RFID sensing with software-defined radio," in *Proc. 16th Int. Conf. Mobile Computing and Networking*, 2010, pp. 97–104.